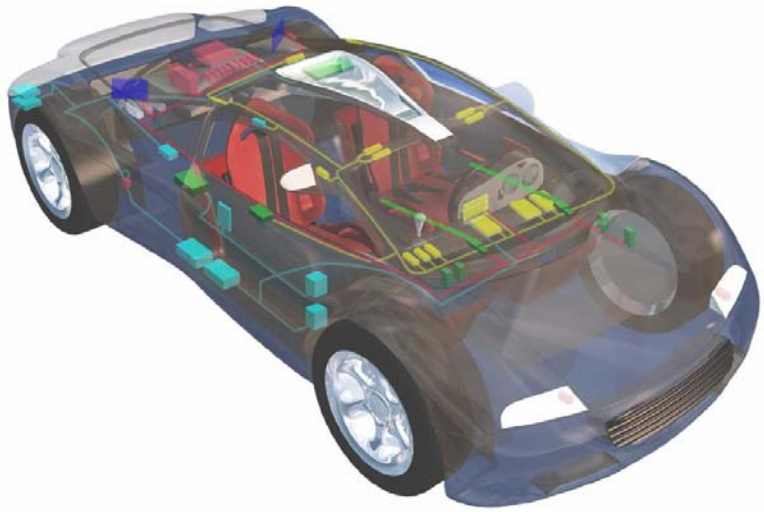
16
Time Triggered Protocol (TTP)



[TTtech04]

18-549 Distributed Embedded Systems

Philip Koopman

October 25, 2004

Significant material drawn from:
Prof. H. Kopetz [Kopetz]
TTP Specification v 1.1 [TTTech]

Carnegie Mellon

Where Are We Now?

- **Where we've been:**
  - Protocol mechanisms & performance
  - CAN – an event-centric protocol with priorities
  - FlexRay – a "flexible" X-by-Wire protocol

- **Where we're going today:**
  - TTP – a TDMA X-by-Wire protocol with additional services

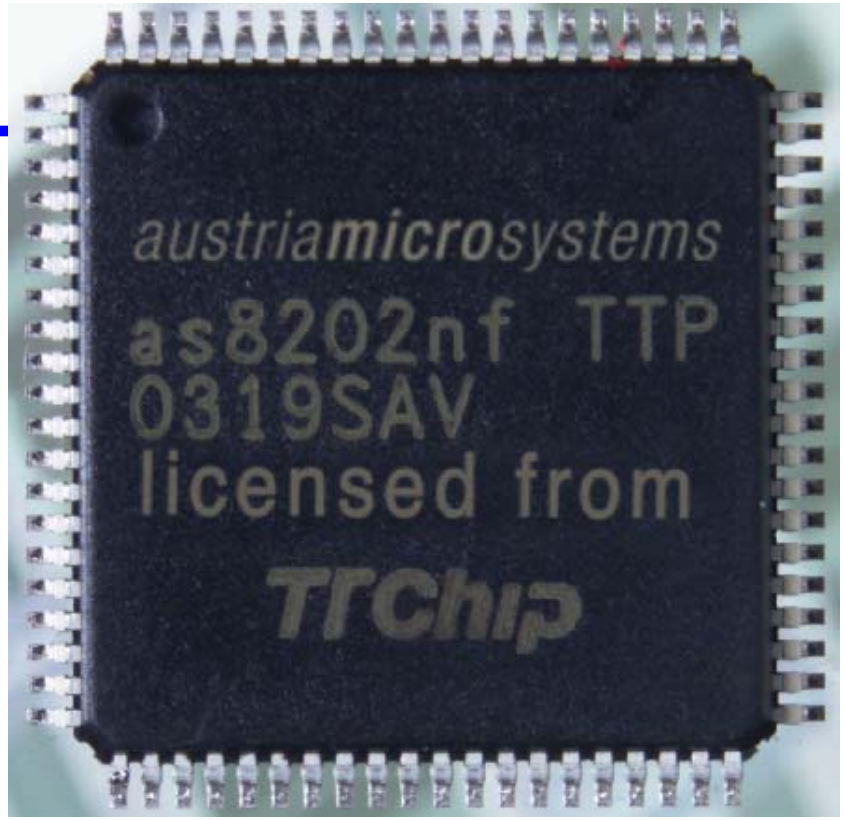
- **Where we're going next:**
  - Test #2 review
  - Test #2

Test #2 – bring a calculator

Preview

TTA = Time Triggered Architecture

TTP = Time Triggered Protocol



[TTTech04]

- **TTP – more than just a protocol**
  - TTP/C Network protocol ("/C" means automotive class C = hard real time)
  - Operating system scheduling philosophy
  - Fault tolerance approach
- **Time-triggered approach**
  - Cyclic schedules
  - Stable time base used to provide access to network (no overt "arbitration")
  - Peer-based system – no master node(s)
  - Also an inexpensive variant (TTP/A) (automotive Class A = soft real time)

TTP History

- **Origins: research at TU Vienna / Prof. Hermann Kopetz**
  - MARS fault tolerance project started in 1979

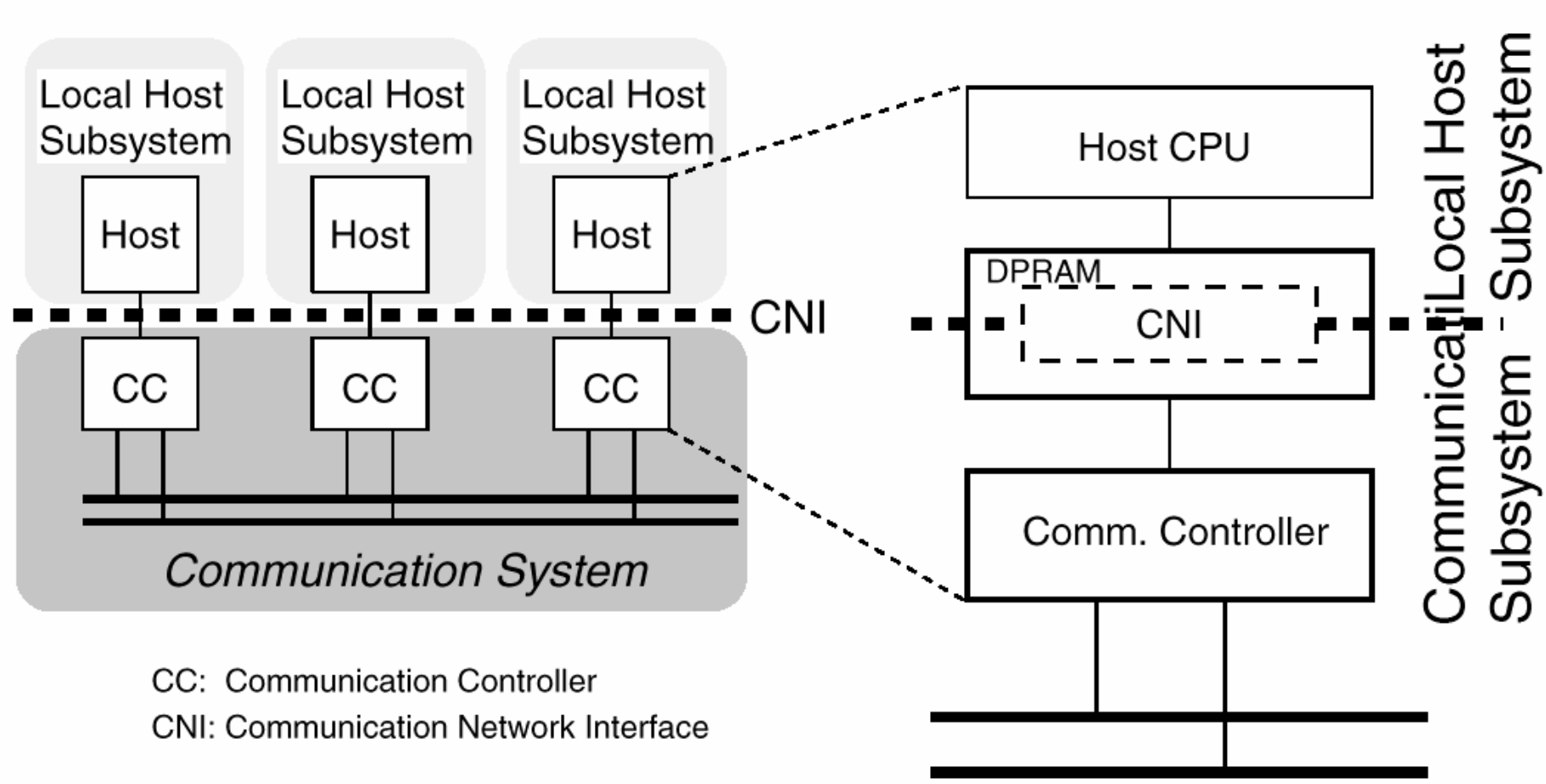
- **Originally designed as "backbone" communication bus for automobiles**
  - First published in 1994
  - Designed for X-by-Wire
  - Safety critical applications

- **Protocol has evolved**
  - Correct problems found in extensive reviews & testing
  - Added higher level services; list stabilized in 1997-1998
  - Add a few more capabilities (some to compete with FlexRay)

- **Also finding a home in other areas**
  - Aviation applications (e.g., Honeywell general aviation flight controls)

TTP Context:

- **TTP/C for Class C automotive applications (critical + hard real time)**
  - Redundant bus for reliability

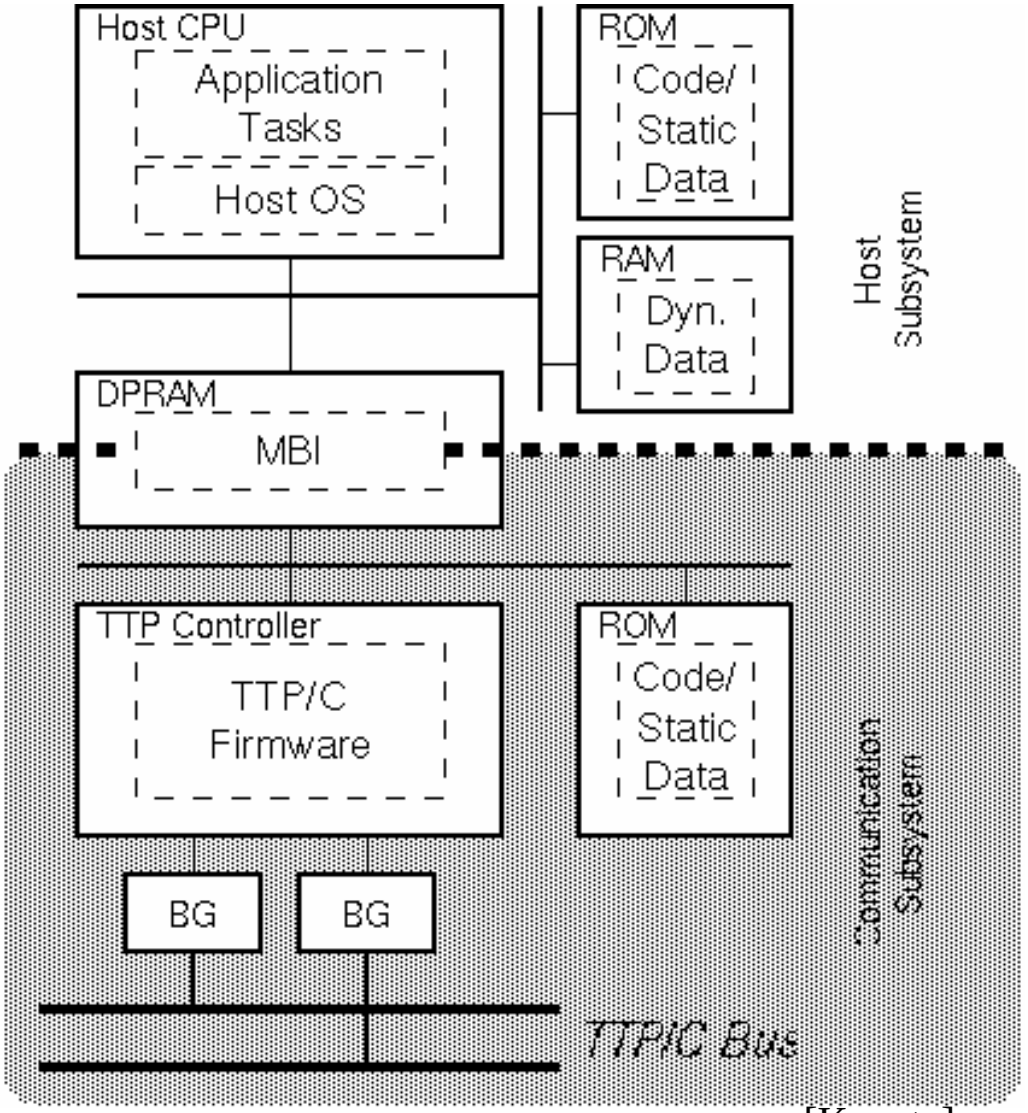


[Most pictures from Kopetz's TTP writings]

Single Node Configuration

- **Includes controller to run protocol**
- **DPRAM = Dual Ported RAM**
  - Used to implement memory-mapped network interface (state variables a.k.a. “mailboxes”)
- **BG = Bus Guard**
  - Hardware watchdog to ensure “fail silent” behavior
  - Guards against “babbling idiots”
- **Real chips must use highly accurate time sources**

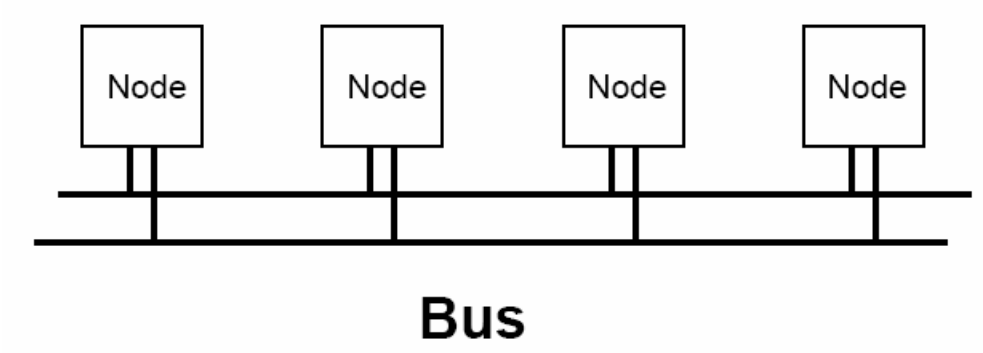
  (Redudant oscillators – one for controller & one for BG)

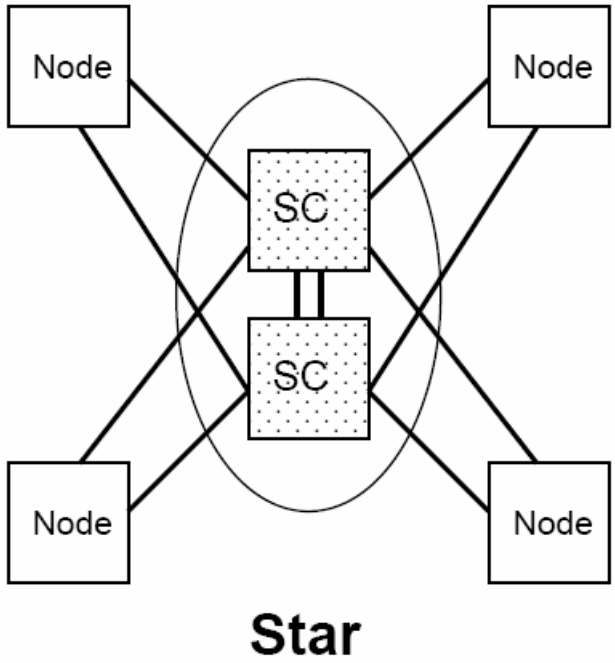


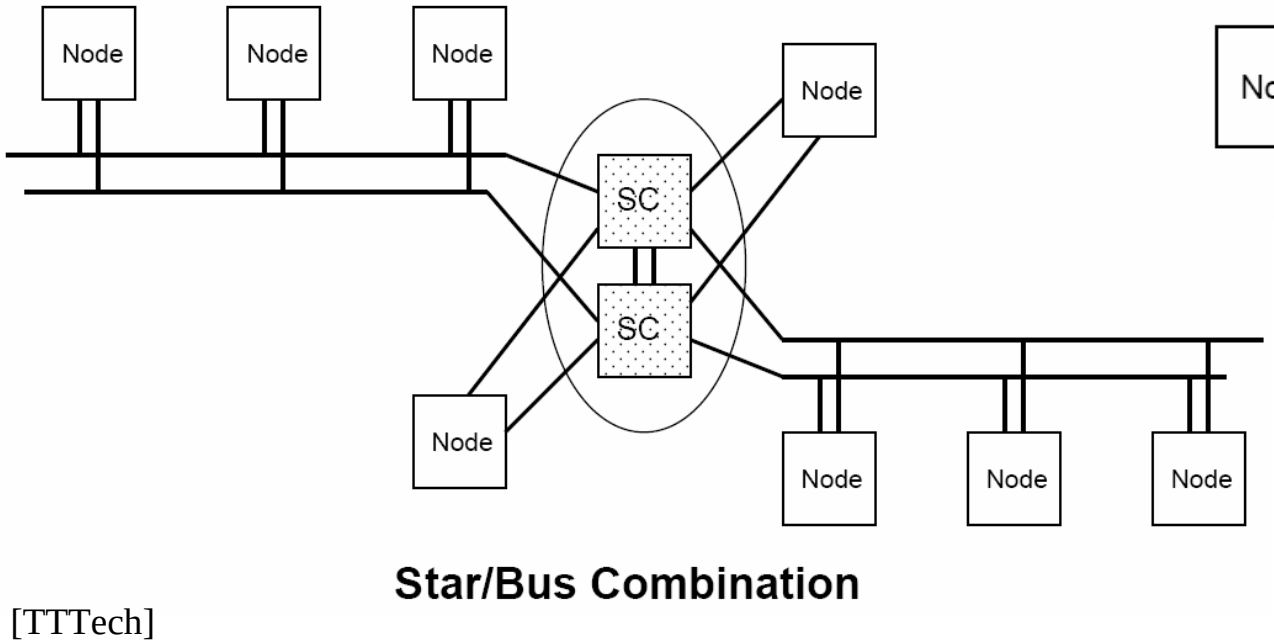
[Kopetz]

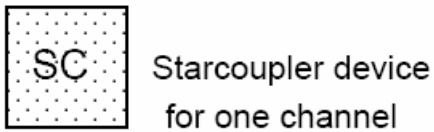
System Topologies

- **Originally just a bus system**
  - Probably need an active hub to do startup properly with some faults









[TTTech]

Computational Clusters & FTUs

- **Cluster = all the nodes on a particular network**
- **FTU = Fault Tolerant Unit = nodes performing identical computations**
  - Assume node fails silent / can use "voting" to determine correct answer

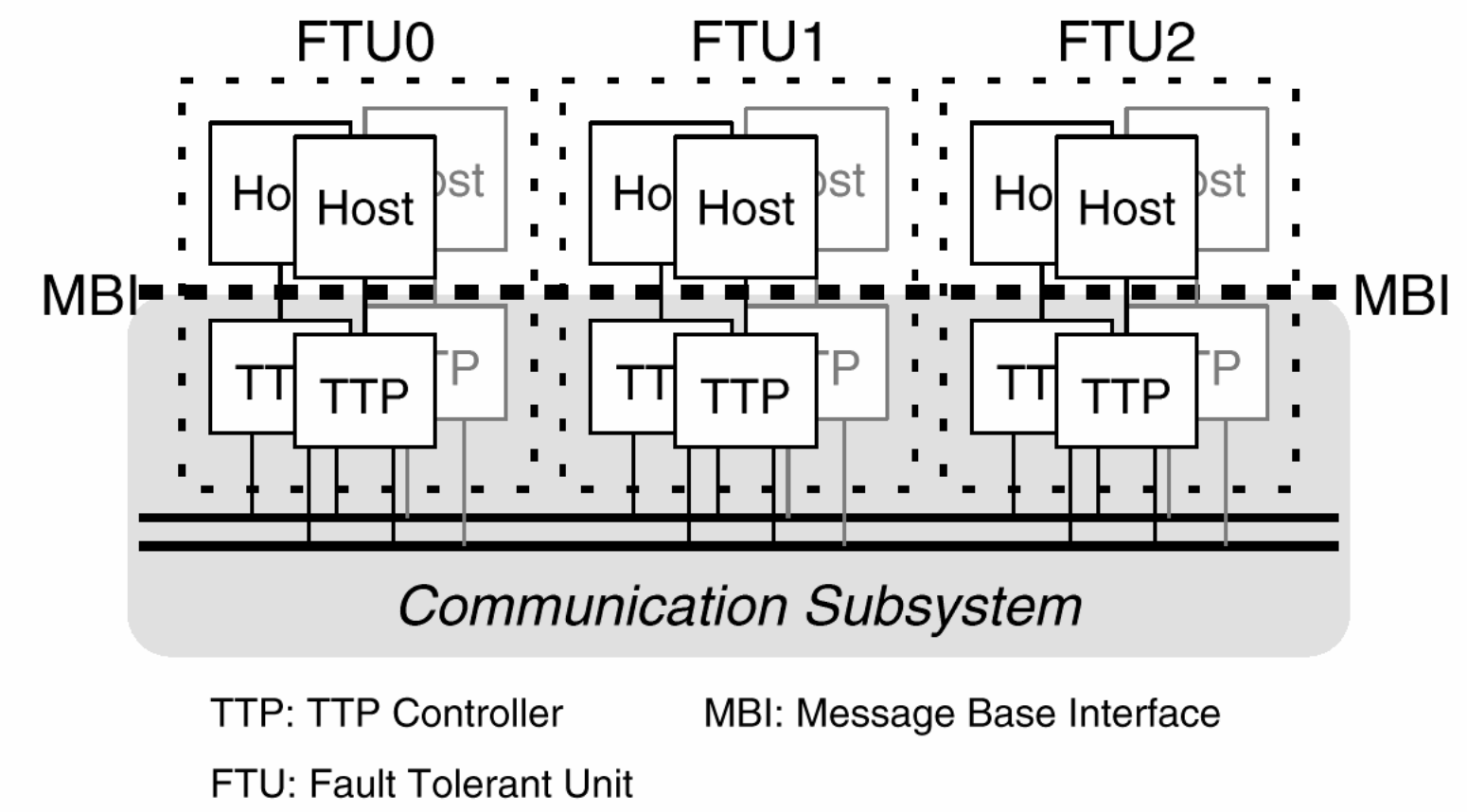
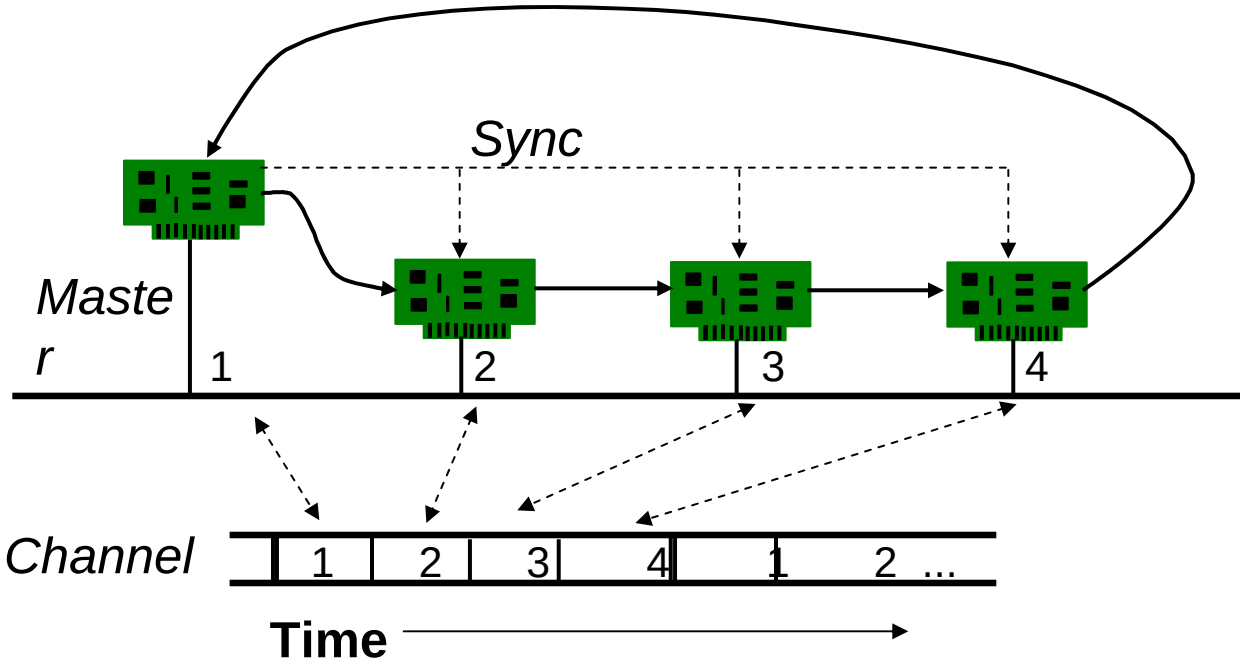


Figure 1: Computational Cluster

[Kopetz]

TDMA - Time Division Multiplexed Access



- **Operation**
  - Master node sends out a frame sync to synchronize clocks
  - Each node transmits during its unique time slot

- **Examples**
  - Satellite Networks, DATAC, TTP

TTP/C Is A TDMA Approach

- **Static schedule for *all* messages in system – predetermined ordering!**
  - A completely deterministic TDMA approach
  - All tasks synchronized to network TDMA schedule as well

- **A TDMA cycle – each FTU gets to compute and broadcast its results in turn**
  - Each FTU sends results twice to reduce problems with lost messages
  - Then next FTU sends some results
  - And so on, coming back to the next message from the first FTU
  - (Does not have to be strict rotation – can be any arbitrary static schedule)
    - But time synchronization works better if things are spread out

- **A "cluster cycle" occurs when all possible messages have been sent**
  - TDMA cycle sends messages from the different nodes
  - Cluster cycle involves scheduling all possible messages + all possible tasks

TTP Cycles

- **TDMA cycle – nodes take turns broadcasting predetermined messages**
- **Cluster cycle – accounts for all possible tasks/messages**

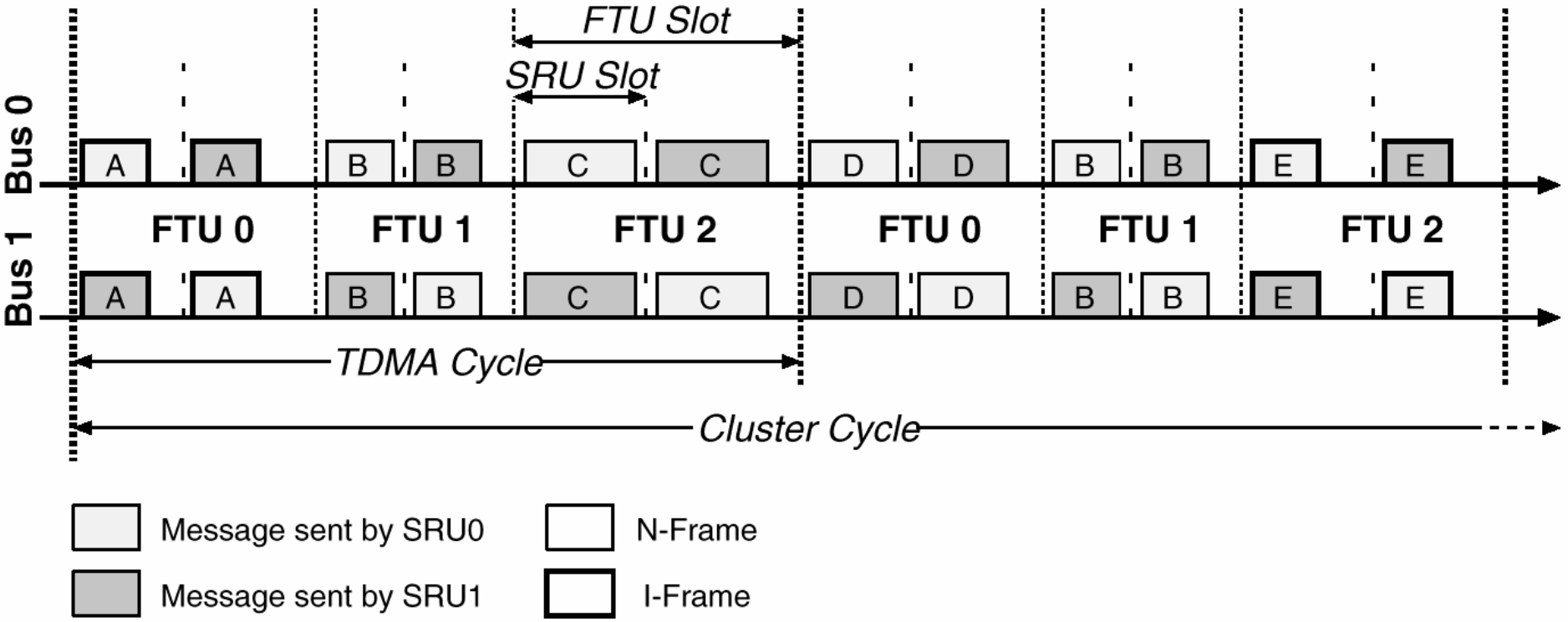


Figure 5: TTP Bus Access Scheme

Dependable Time Sources Are Important

- **Maybe even dual redundant crystal oscillators/DATAC for Boeing 777**

- **Example: TTP AS8202 requires two oscillators**
  - One for main chip oscillator
  - One for independent bus guardian

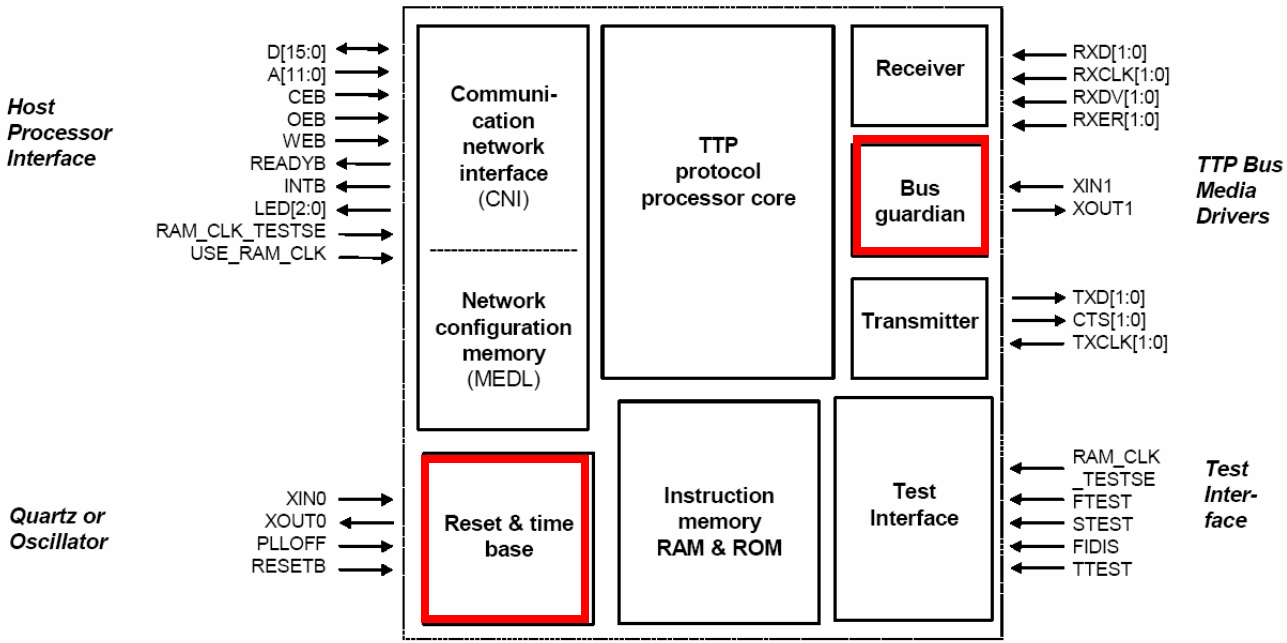
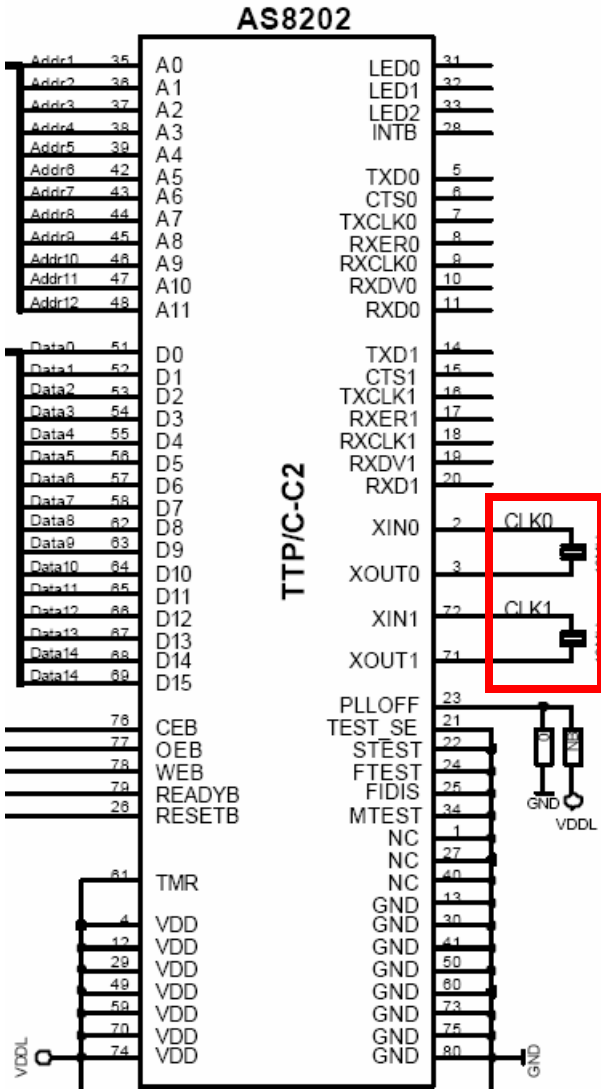


Figure 1. AS8202NF Block Diagram



MeDL – Message Descriptor List

- **Globally shared schedule of messages and ordering**
  - All nodes know the entire message schedule
  - Only one node is permitted to transmit at a time, and only its predetermined message
- **Static schedule simplifies arbitration and resynchronization**
  - Every node knows when its turn is based on time, so no arbitration is required
  - Designer can allocate appropriate compute time to avoid receiver over-runs
  - If a message is missed, all nodes know what was missed by when it failed to arrive
- **But, some challenges**
  - Requires stable time sources
    - Nodes adjust every time a message arrives based on knowing expected time from MeDL for that message
    - Requires arbitration to start network and add nodes

N-Frames & I-Frames

- **I-frames used for initialization**
  - Also sent occasionally to permit recovered nodes to resync to cluster
  - C-state is current state of system (time & position in cluster schedule)
- **N-frames for normal messages**

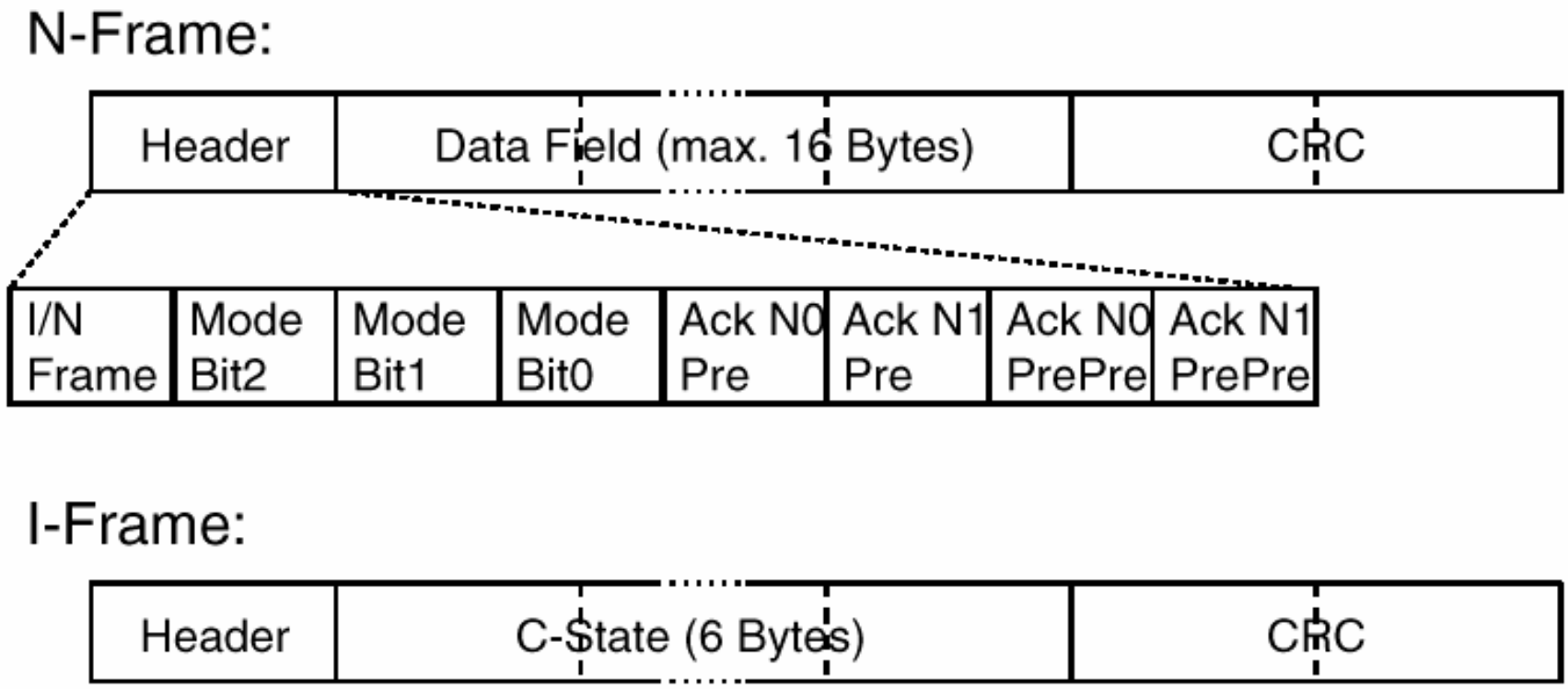


Figure 4: TTP Frame Formats

[Kopetz]

Why Consistency & Group Membership?

- **Fundamental distributed system problem – agreement**
  - How can you make sure that all other nodes get your message?
  - How do you know when all other nodes actually got your message?
  - Provably impossible with asynchronous system
  - Requires very tricky algorithms and some notion of a “timeout” or time triggered system

- **FlexRay approach – application has to deal with this**

- **CAN approach – ack + Nack multicast acknowledge**
  - Provides partial solution – doesn’t detect dead/offline nodes by itself

- **TTP – group membership**
  - All nodes in your group have seen same messages you have seen
  - If messages diverge, then groups split in a short bounded time
  - Thus, if a node is still in your group, you know it got your message*

Simplified Look At Group Membership

- **C-state is internal state of the TTP/C controller**
  - Global time value
  - Current slot in cluster cycle
  - Cluster mode (mode changes permit changing MEDL)
  - Membership information (which nodes are in the current group)

- **A node sending you data is in your group if:**
  - You've received a correctly formatted message (passes CRC data integrity test)
  - That node's C-state matches your own C-state (i.e., you both agree on protocol state)

- **TTP/C approach to sending C-state**
  - Include C-state in computed message CRC, but don't actually send the bits
  - Thus, message CRC only checks out OK on receiver if C-states match
  - If node isn't in your group or diverges, you simply ignore its messages (because messages will fail CRC checks)

TTP Design Principles

- **Consistent Distributed Computing Platform**
  - All correct nodes have exactly the same state (replica determinism)
  - Any node that doesn't have same values of state variables is ejected from group
- **Temporal "firewalls"**
  - Pure time triggered design – no node can affect timing of rest of system directly
- **Composability** (If you have enough slack in TDMA schedule)
  - Changing a node or message doesn't disturb other nodes
  - Adding a node or message doesn't disturb other nodes
- **Fault Tolerance as a built-in service**
  - Fault Hypothesis: any single component suffers arbitrary failure
  - Assume that error detection takes place before second failure can occur
  - Assume controller & cluster design are free of design faults
- **Scalability**
  - Pure time triggered/state variable approach said to promote scalability

Other TTP/C Services

- **Fault tolerant global time base**
  - Precision in the microsecond range to all nodes without time master
- **Consistent membership service**
  - Each node updates itself about state of other nodes within two TDMA rounds
  - Distributed agreement algorithm – only nodes in complete agreement in a group
- **Clique avoidance**
  - Prevents fragmentation of network into multiple competing groups
- **Arbitrary single hardware failure tolerated**
  - This includes testing with radiation-induced arbitrary faults
- **Protection from maliciously faulty host**
  - TTP/C chip (with bus guardian) guarantees host can’t kill protocol operation

Fault Tolerance Approach Based On Redundancy

- **Send each message twice**
  - Assume random bit errors don't happen back-to-back
  - Avoids need for acknowledgements (most of the time…)
- **Have two (or more) sets of hardware**
  - Redundant sets of hardware send extra messages
  - Distributed, fault-tolerant time master
  - When one set fails, backup automatically intervenes

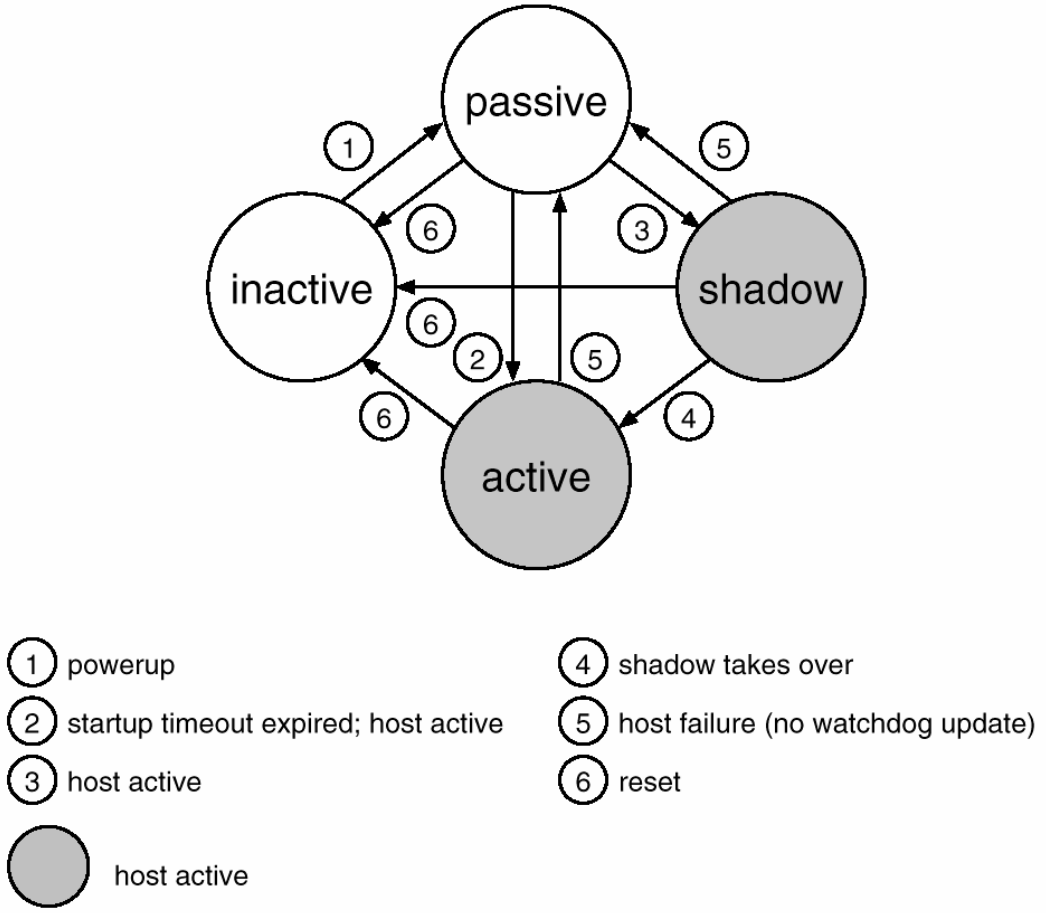


Figure 7: Activity State Changes

[Kopetz] 2:

FTU Configurations

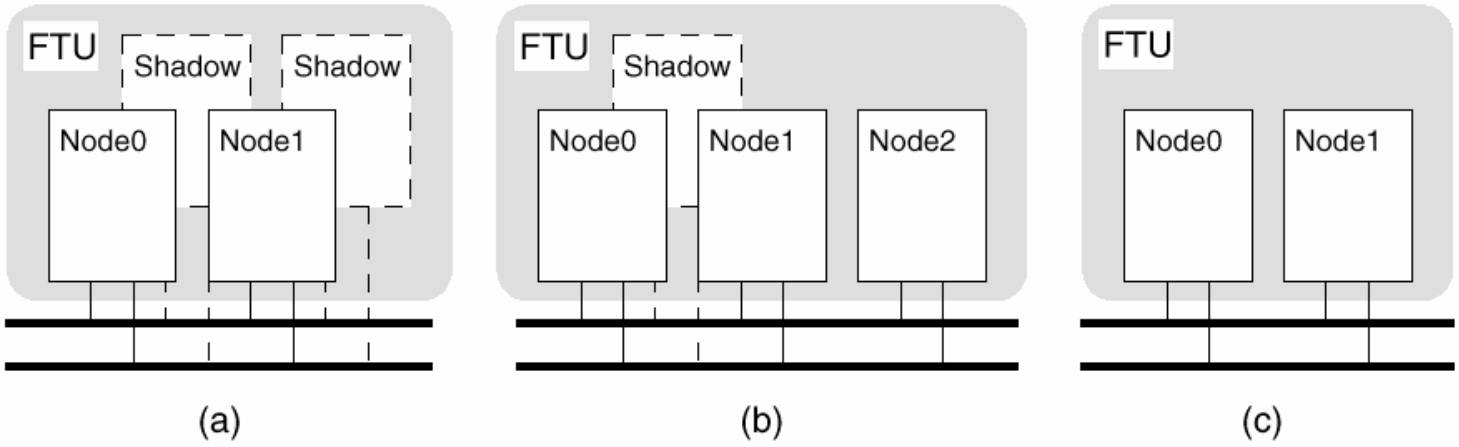
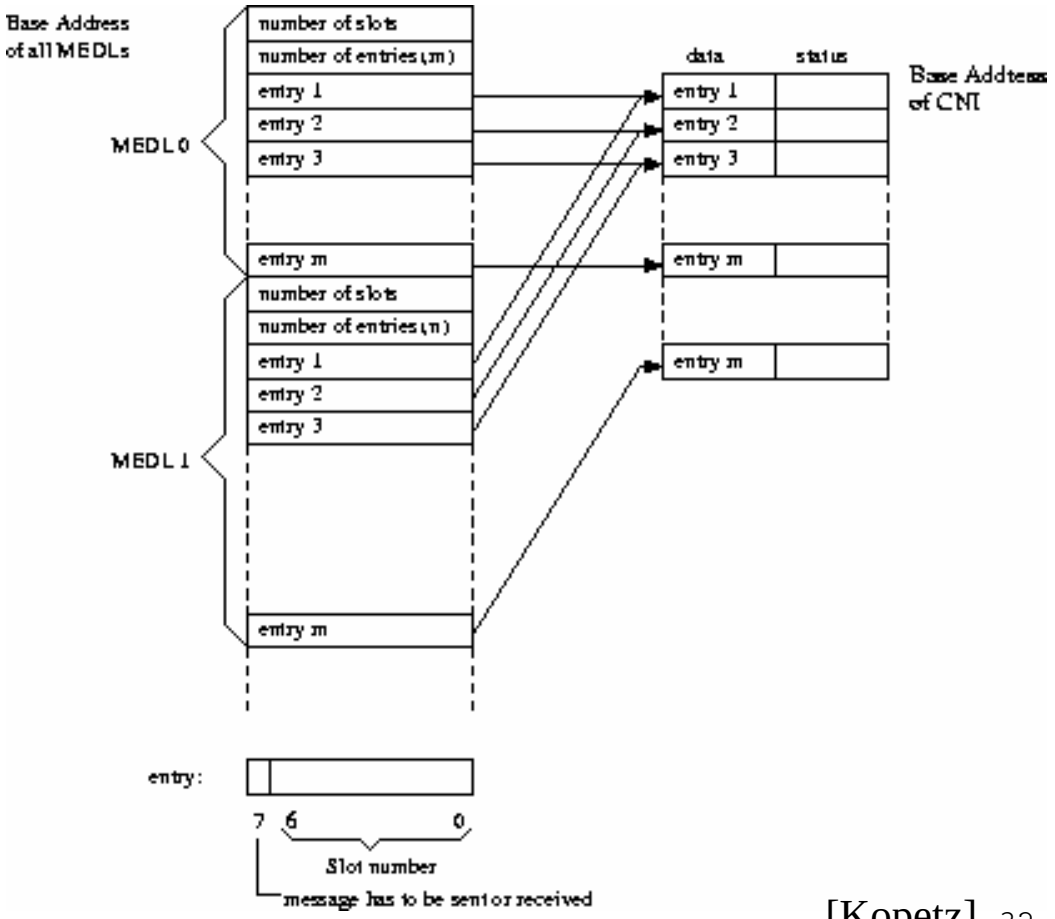


Figure 2: FTU Configuration Examples: (a) Two active nodes, two shadow nodes; (b) Three active nodes (Triple Modular Redundancy) with one shadow node; (c) Two active nodes without a shadow node

[Kopetz]

Operating Modes

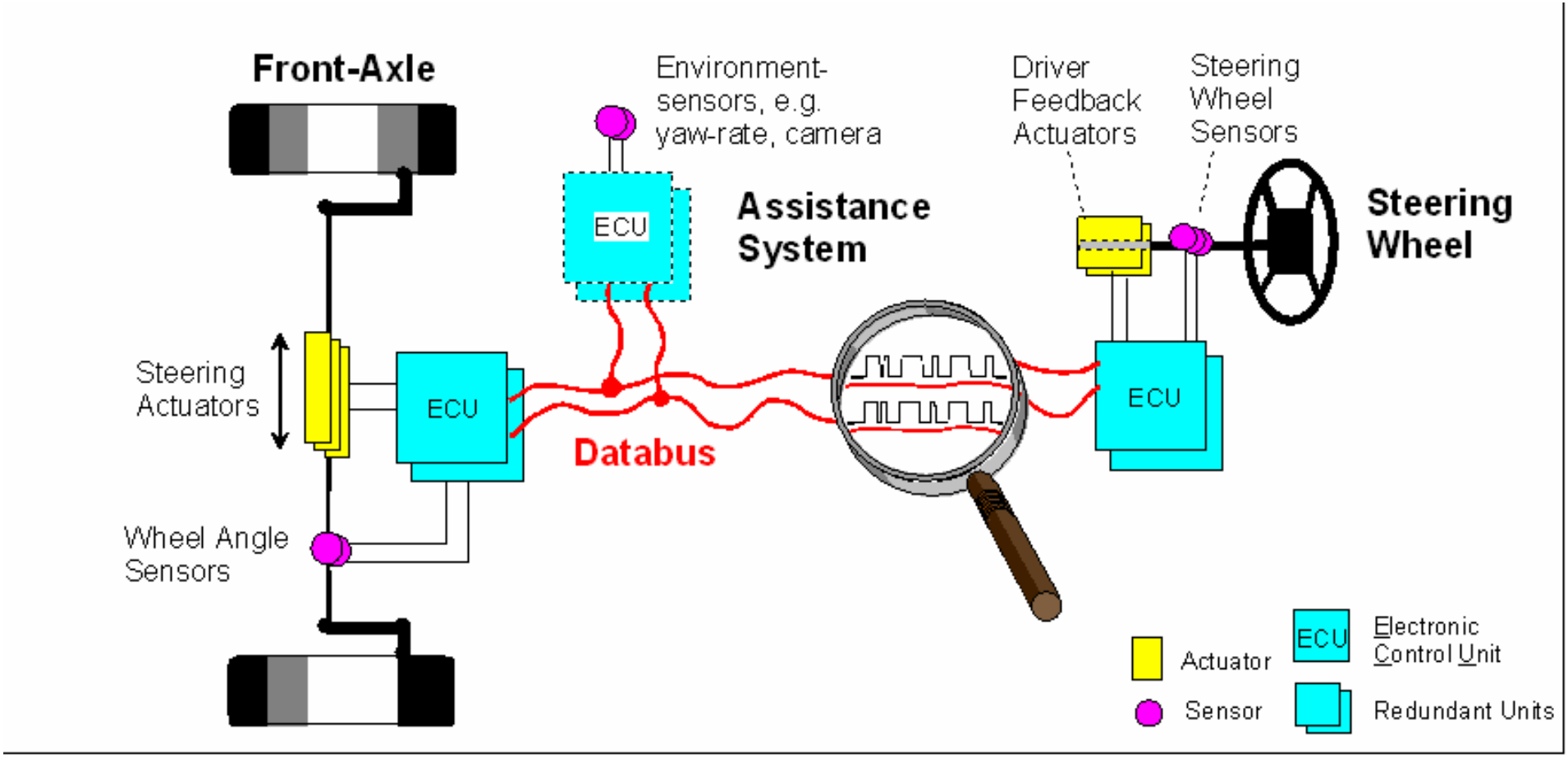
- **Different operating modes require different message schedules**
  - Accelerating vs. cruise might need different information
  - Operation vs. diagnosis need emphasis on different aspects of the vehicle
  - Failure recovery might need access to different message traffic
- **TTP solution: use multiple schedules**
  - Precompute a different MeDL for every possible situation
  - (And invent tool support to make this feasible)
- **Basis of Mode selection for TTP/C**



[Kopetz]

Advanced Application: X-by-Wire

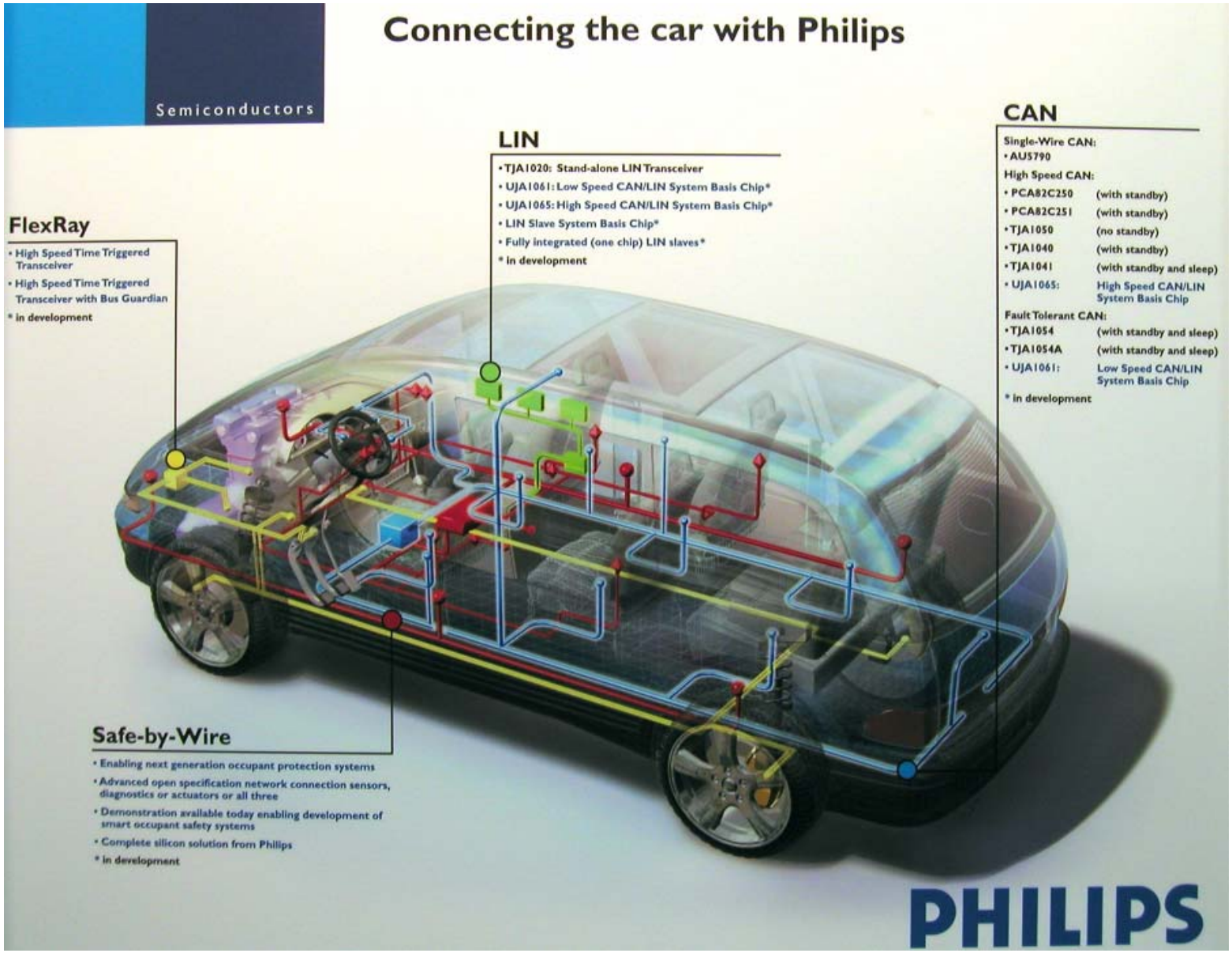
- **Look Ma – no mechanical connection!**
  - Digital (or optical) connections between user controls and actuators
  - Airplanes do it now, so why not cars?



[Kopetz]

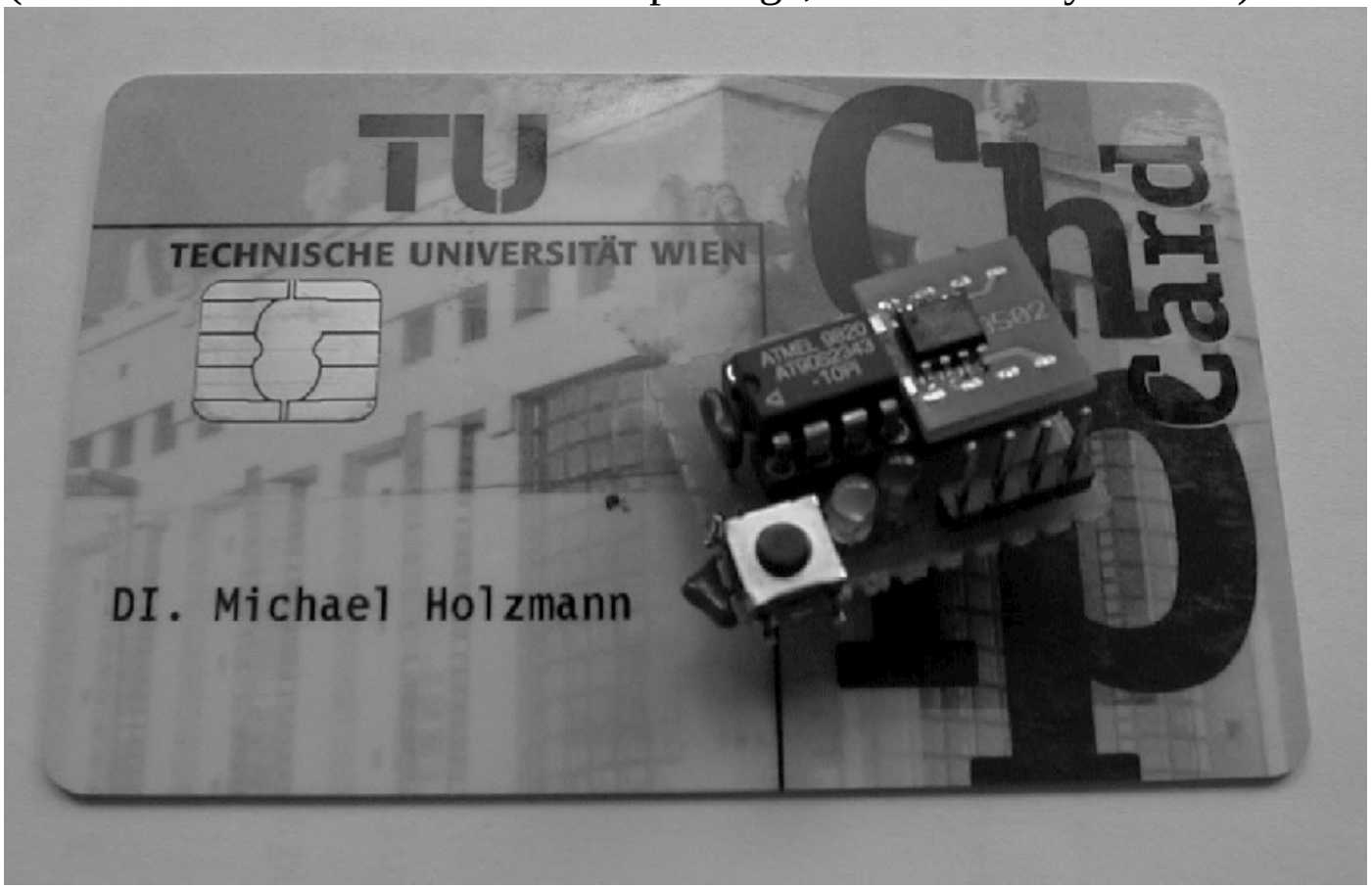
But, There Are Multiple Networks In A Vehicle

- **Some applications are less demanding – and more cost sensitive**



TTP/A: A Reduced Cost Version

- **How do you do this for about $2 per node?**
  - Answer: you make compromises… and use on Class A devices (soft real time)
  - Distributed fault tolerance is expensive (especially time bases), so go master-based TDMA instead
    - (TTP folks call this “master slave polling”, but it is really TDMA)



[Kopetz]

TTA = Time Triggered Architecture

- **TTP/A operates in lock step with cluster's TTP/C schedule**

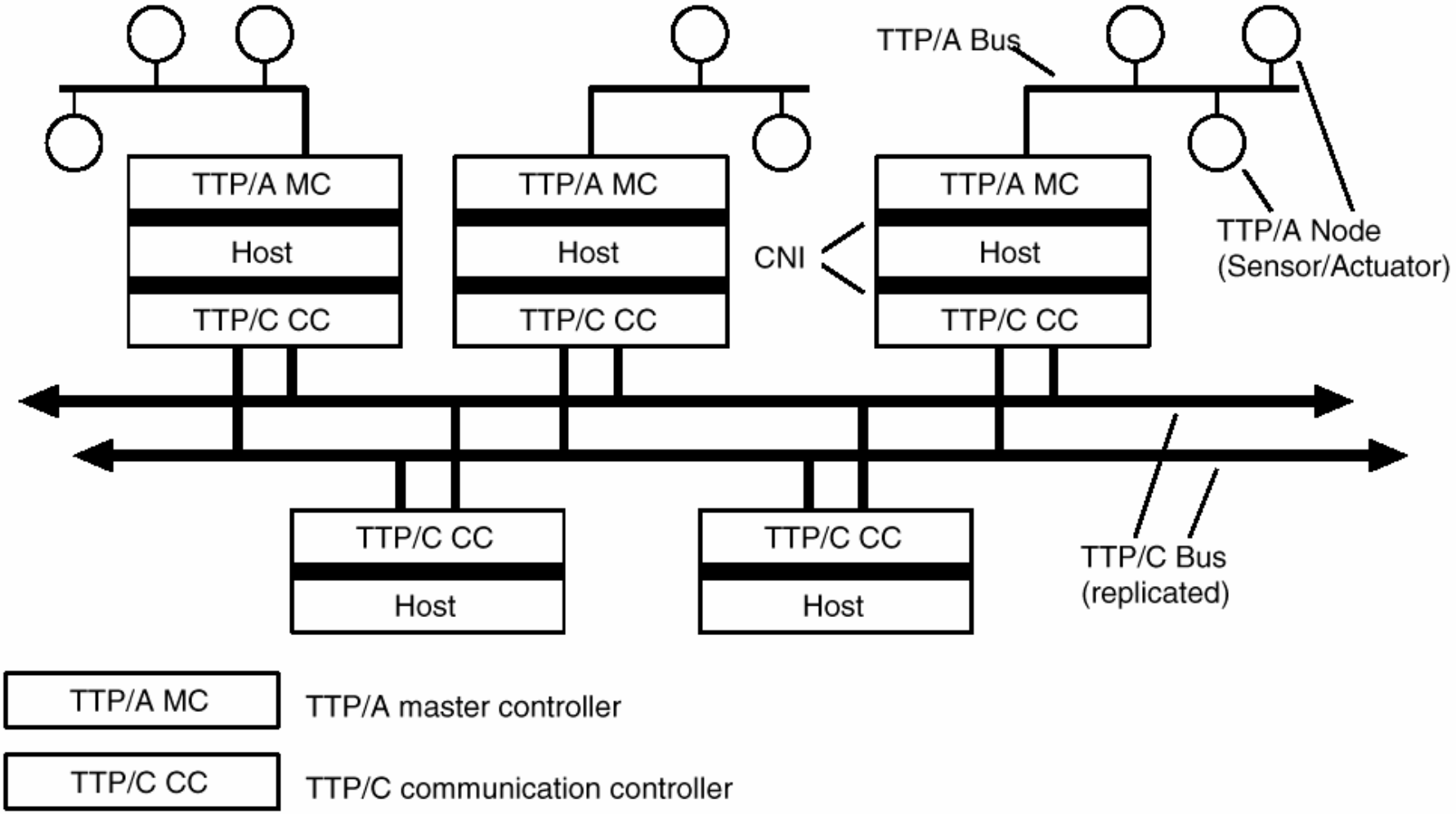


Figure 1: A TTA cluster with three fieldbusses

[Kopetz]

TDMA With Very Short Message Chunks

- **Use master node instead of implicit agreement/I-frame distribution of bus timing responsibility**
  - Each frame is one byte(!)

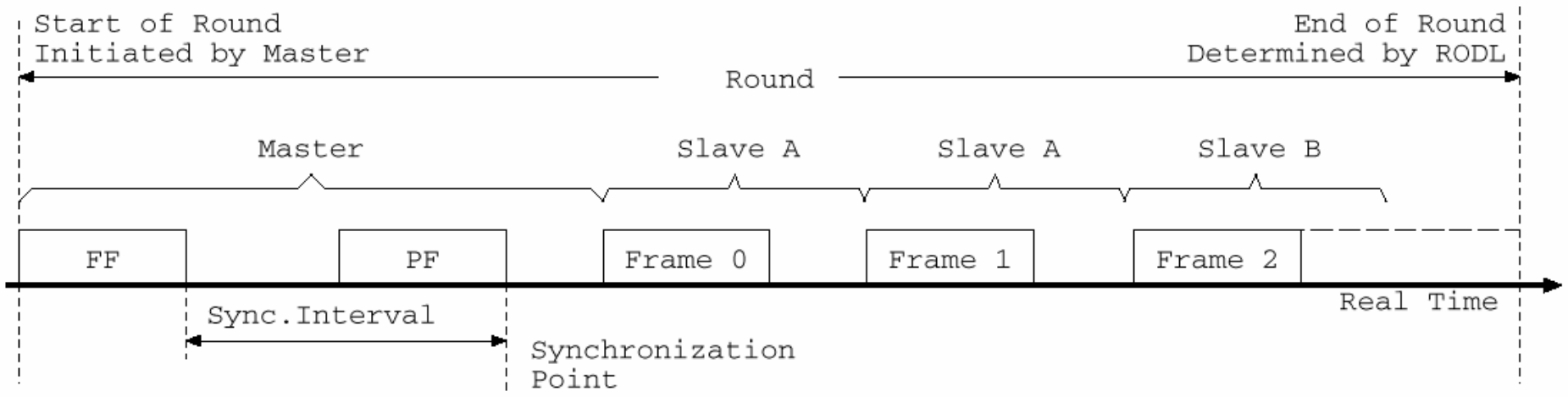
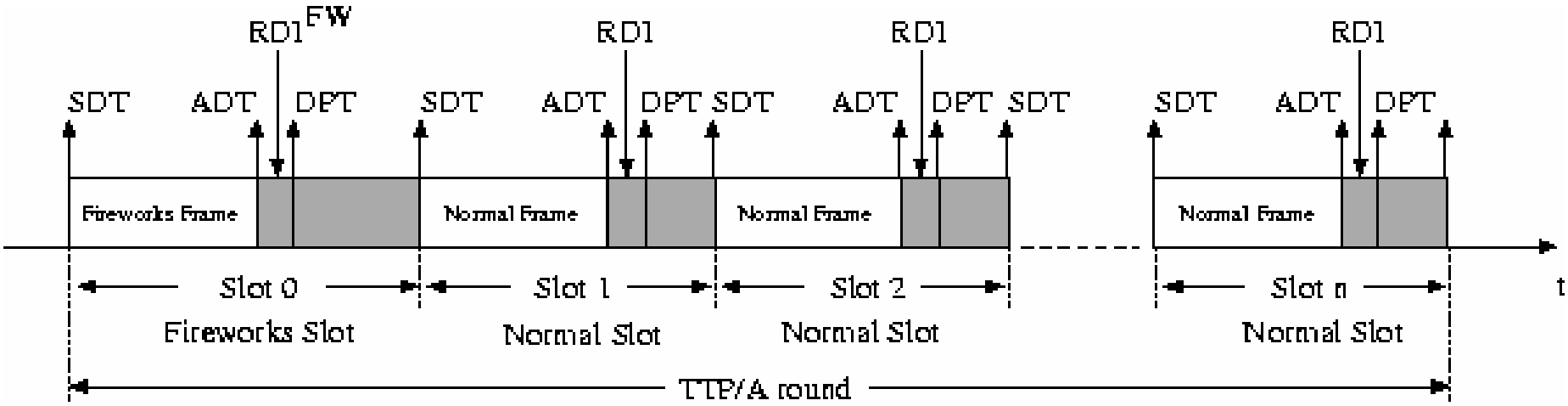


Figure 2: Structure of a TTP/A round

But TTP/A Is Suppose To Be Cheap…

- **… so each message frame is only a byte long within the round**

- **"Fireworks" Frame used by master to denote start of round**
  - Includes MeDL number for system reconfiguration to new modes



[Kopetz]

TTP/A Sensor Redundancy

- **The "dependable" way**

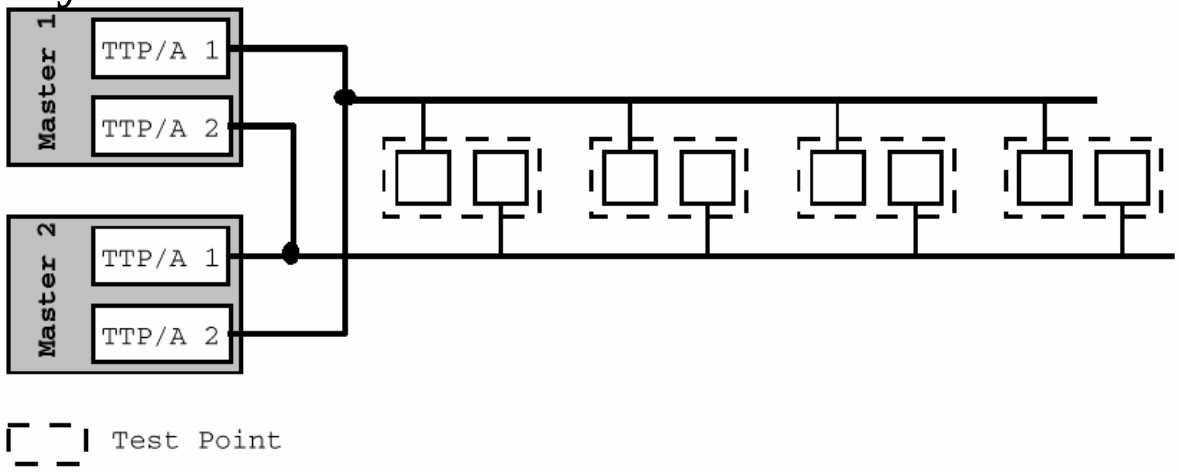


Figure 5: Sensor Replication in TTP/A

- **The "cheap" way**

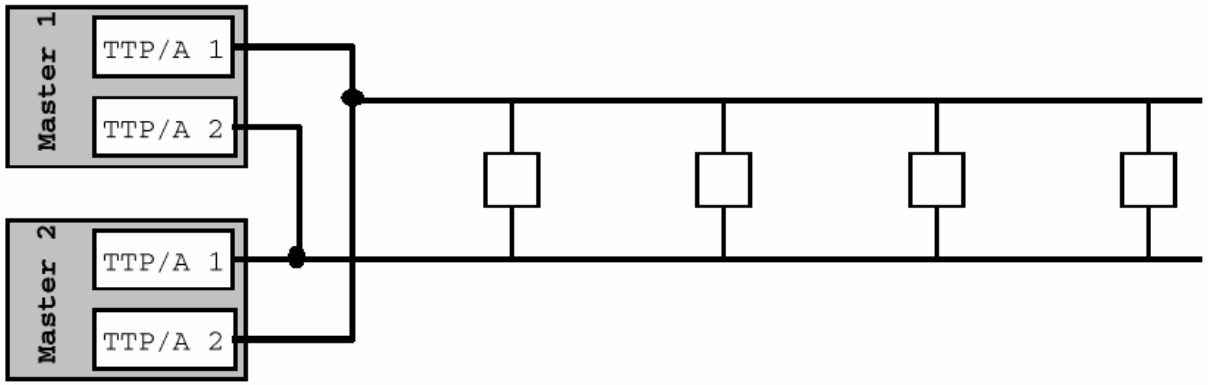
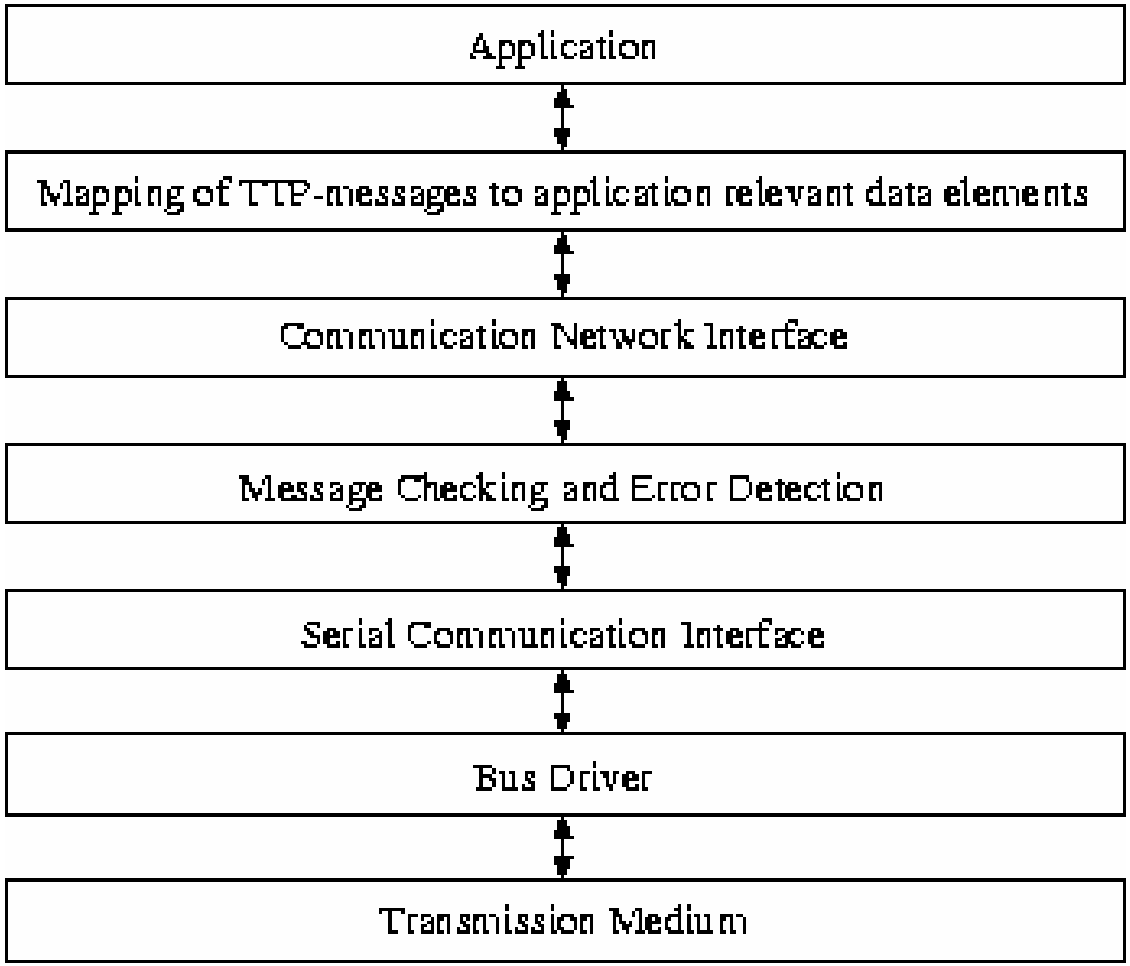


Figure 4: Use of Replicated Busses in TTP/A

[Kopetz]

TTP Is A "Total" Approach

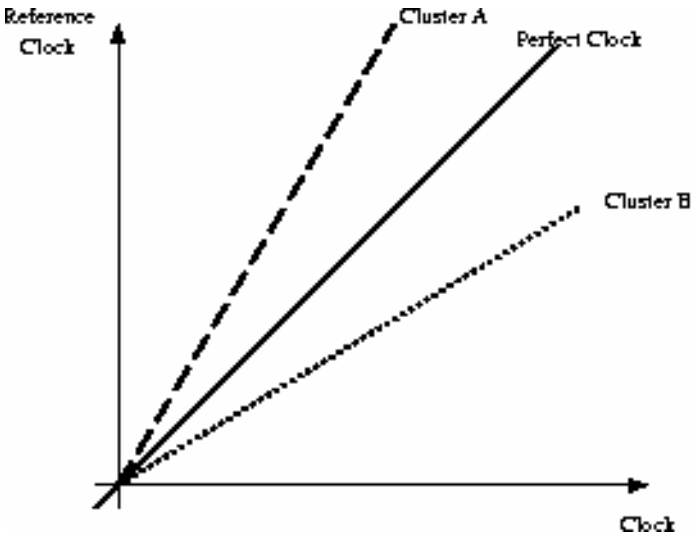
- **Goes from Application scheduling to network**
  - Defines message construction
  - Defines network arbitration
  - Defines task execution times
- **Static system scheduling**
  - Multiple schedules for mode shifts

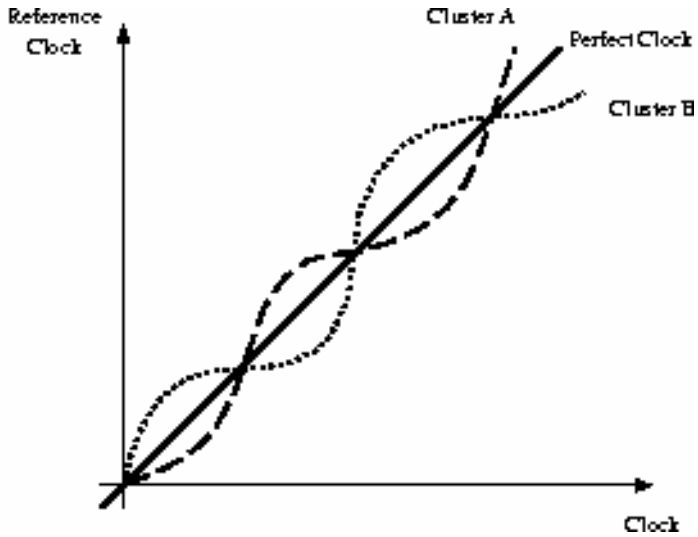


[Kopetz]

Time Synchronization

- **Clusters resynchronize over time**
  - Important that differences be unbiased for this to work





[Kopetz]

Tradeoffs (adjusted for TTP/A)

- **Advantages**
  - Simple protocol to implement;
    historically very popular
  - Bounded latency for real-time applications

- **Disadvantages**
  - Single point of failure from centralized master
  - Polling consumes bandwidth
  - Network size fixed during installation (not robust)
    - Or, master must discover nodes during reconfiguration
    - Or, master has to modify MEDL for each reconfiguration
  - Static scheduling – no prioritization
    - But, can use centralized load balancing

Tradeoffs (adjusted for TTP/C)

- **Advantages**
  - Simple protocol to implement
  - Deterministic response time
  - No wasted time for Master polling messages

- **Disadvantages**
  - Single point of failure from the bus master – *So TTP uses distributed mastership*
  - Wasted bandwidth when some nodes are idle (*or when values don't change*)
  - Requires stable clocks
  - Network size fixed during installation (not robust) – *I-nodes permit admissions*
  - Prioritization is static – *TTP doesn't use priorities at all*

US005694542A

United States Patent [19]

Kopetz

[11] Patent Number: 5,694,542

[45] Date of Patent: Dec. 2, 1997

[54] **TIME-TRIGGERED COMMUNICATION CONTROL UNIT AND COMMUNICATION METHOD**

[75] Inventor: **Hermann Kopetz**, Baden, Austria

[73] Assignee: **Fault Tolerant Systems FTS-Computertechnik Ges.m.b.**, Baden bei Wien, Austria

[21] Appl. No.: **562,337**

[22] Filed: **Nov. 24, 1995**

[51] **Int. Cl.**[6] **G06F 11/00**; G06F 11/10

[52] **U.S. Cl.** **395/185.02**; 395/185.03; 395/185.04; 395/185.05; 395/184.01

[58] **Field of Search** 395/182.08, 185.01, 395/185.02, 185.03, 185.04, 185.05, 185.06, 185.07, 184.01, 183.07, 182.16, 200.11, 200.13; 364/264, 266.3; 455/9, 39, 67.1, 115

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,241,398	12/1980	Carll	395/200.17
4,534,023	8/1985	Peck et al.	370/384
4,598,404	7/1986	Perry et al.	371/49.1
4,646,298	2/1987	Laws et al.	371/10.2
4,646,300	2/1987	Goodman et al.	371/33
4,947,484	8/1990	Twitty et al.	371/37.1
4,956,641	9/1990	Matai et al.	340/825.44
4,980,906	12/1990	Forson et al.	379/32
5,428,629	6/1995	Gutman et al.	371/37.1

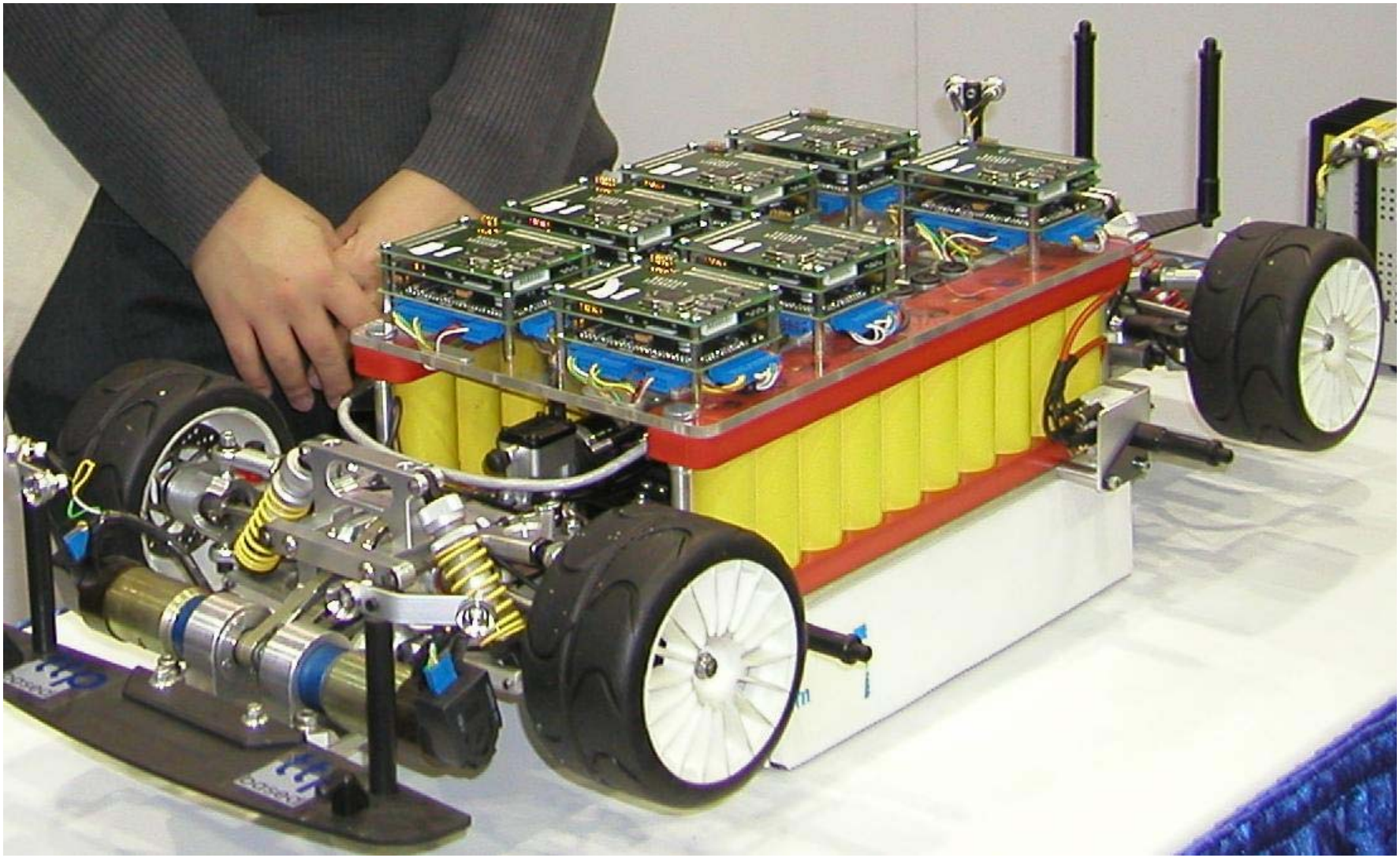
Primary Examiner—Hoa T. Nguyen
Attorney, Agent, or Firm—Thomas R. Vigil

[57] **ABSTRACT**

The time-triggered communication control unit and method for the transmission of messages within a distributed real time computer architecture consist of a plurality of fault-tolerant units, in which the information to be transmitted consists of a control field (K), a data field (D) and a CRC (Cyclic Redundancy Check) field (CRC). The contents of the CRC field are calculated over the concatenation of the control field (K), the data field (D) and the local internal state of the transmitting communication control unit. The local internal state of such a communication control unit consists of a concatenation of the global time with a membership field. The membership field is composed of a sequence of bits, where each bit is assigned to a unique fault-tolerant unit. The value TRUE of a membership bit means that the assigned fault-tolerant unit is operating, the value FALSE means that it is faulty. By recalculating the CRC with its internal state, the receiving communication control unit can recognize both incorrect information or a difference between the internal states of the transmitting and the receiving communication control unit.

15 Claims, 5 Drawing Sheets

TTP/C Demonstrator Vehicle (2000)



How Do You Know A Protocol Is OK?

- **TTP is probably most-studied X-by-wire protocol**
  - And, after years of development, new issues are being revealed
  - Most of the issues are relatively minor – indicates this is a difficult problem
  - Expect other protocols (e.g., FlexRay) to go through a similar process

- **Main techniques for TTP Validation:**
  - Careful design
  - Early publication of details
  - Formal verification of algorithms
  - Extensive testing
  - Physical fault injection

[TTTech04]

Review

- **TTP – more than just a protocol**
  - Network protocol
  - Operating system scheduling philosophy
  - Fault tolerance approach

- **Time-triggered approach TTP/C**
  - Cyclic schedules
  - Stable time base used to provide access to network (no overt “arbitration”)
  - Very simple to implement the usual stuff
    - Startup is painful
    - Mode shifts are painful
    - Stable time base is painful
  - Also a cheaper master/slave variant…TTP/A